Generalized Formalization of Games

Christiaan van de Sande Tanner Reese

September 23, 2020

1 Introduction

2 Games

3 Rules

3.1 Motivation and Definition

The purpose of a rule is to define legal transitions between states in a game. It may be tempting to define a rule as a function from one state to another state, but this approach is very limited. It does not help to describe the moves themselves, only the consequences of those moves. The definition of a rule must include the definition of a function λ that generates the legal moves from a given state and the definition of a function ϕ that executes a given move on a given state. The moves must be independent of the state, so that the same move can be executed on different states. Moves must also be reversable, meaning that, given any states s and s' and move m, where m is the move from s to s', there must be a way to find s from only s' and m. This reverse operation is the responsibility of a third function ρ. The requirements of these three functions gives the following definition for a rule:

Definition 3.1.1. *r is a* **rule** *on sets A and B (written $r \in \mathcal{R}(A, B)$) iff $\lambda_r : A \to \mathcal{P}(B)$, $\phi_r : A \times B \to A$, $\rho_r : A \times B \to A$, such that $\phi_r(\rho_r(a, b), b) = a$ and $\rho_r(\phi_r(a, b), b) = a$.*

For the rule-of-play, r, of a game, the set of all well-formed states, S, forms an input set for r and the set of all well-formed moves, M, forms an output set for r, so $r \in \mathcal{R}(S, M)$. However, not all rules have the set of well-formed states as their input set. Take, for example, the rule for capturing in chess. In most cases, captures are performed by moving one piece onto a square occupied by another piece (the notable exception to this rule is en-passant capturing) Thus, the legal captures are dependent, not only on the state (i.e. where all the pieces are located), but also on the move being made, a piece only captures on the square that it moves to. It is *possible* to define a function that finds all legal captures and a second function that finds all legal moves that are not captures, but this approach sacrifices the modularity of having one rule that is responsible for the *movement* of pieces and having another rule that is responsible for the conditions under which pieces are *captured*. For chess captures, then, the input set is the cartesian product of the set of all well-formed states and the set of all piece moves.

3.2 Operations on Rules

Definition 3.2.1. *Let $r \in \mathcal{R}(A, B)$ and s be the* **reduction** *of r (written $s = |r|$). Then $s \in \mathcal{R}(A, A)$, where:*

$$\lambda_s(a) = \begin{cases} \varnothing & \lambda_r(a) = \varnothing \\ \{a\} & \lambda_r(a) \neq \varnothing \end{cases} \tag{1}$$

$$\phi_s(a, b) = a \tag{2}$$

$$\rho_s(a, b) = a. \tag{3}$$

Definition 3.2.2. *Let $r \in \mathcal{R}(A, A)$ and s be the* **negation** *of r (written $s = \overline{r}$). Then $s \in \mathcal{R}(A, A)$, where:*

$$\lambda_s(a) = \begin{cases} \varnothing & \lambda_r(a) \neq \varnothing \\ \{a\} & \lambda_r(a) = \varnothing \end{cases} \tag{4}$$

$$\phi_s(a, b) = \phi_r(a, b) \tag{5}$$

$$\rho_s(a, b) = \rho_r(a, b). \tag{6}$$

Definition 3.2.3. Let $r \in \mathcal{R}(A,B)$ and $s \in \mathcal{R}(A,B)$ and t be the **union** of s and r (written $t = s \cup r$). Then $t \in \mathcal{R}(A,B)$, where:

$$\lambda_t(a) = \lambda_r(a) \cup \lambda_s(a) \tag{7}$$
$$\phi_t(a,b) = \phi_s(a,b) \tag{8}$$
$$\rho_t(a,b) = \rho_s(a,b). \tag{9}$$

Definition 3.2.4. Let $r \in \mathcal{R}(A,B)$ and $s \in \mathcal{R}(A,B)$ and t be the **intersection** of s and r (written $t = s \cap r$). Then $t \in \mathcal{R}(A,B)$, where:

$$\lambda_t(a) = \lambda_r(a) \cap \lambda_s(a) \tag{10}$$
$$\phi_t(a,b) = \phi_r(a,b) \tag{11}$$
$$\rho_t(a,b) = \rho_r(a,b). \tag{12}$$

Definition 3.2.5. Let $r \in \mathcal{R}(A,B)$ and $s \in \mathcal{R}(A,C)$ and t be the **independent product** of s and r (written $t = s \times r$). Then $t \in \mathcal{R}(A, B \times C)$, where:

$$\lambda_t(a) = \lambda_r(a) \times \lambda_s(a) \tag{13}$$
$$\phi_t(a,(b,c)) = \phi_r(\phi_s(a,c),b) \tag{14}$$
$$\rho_t(a,(b,c)) = \rho_s(\rho_r(a,b),c). \tag{15}$$

Definition 3.2.6. Let $r \in \mathcal{R}(A,B)$ and $s \in \mathcal{R}(B,C)$ and t be the **dependent product** of s and r (written $t = s \cdot r$). Then $t \in \mathcal{R}(A, B \times C)$, where:

$$\lambda_t(a) = \{(b,c) | b \in \lambda_r(a), c \in \lambda_s(b)\} \tag{16}$$
$$\phi_t(a,(b,c)) = \phi_r(a, \phi_s(b,c)) \tag{17}$$
$$\rho_t(a,(b,c)) = \rho_r(a, \rho_s(b,b)). \tag{18}$$

Definition 3.2.7. Let $r \in \mathcal{R}(B,B)$ and $s \in \mathcal{R}(A,B)$ and $t \in \mathcal{R}(A,C)$ and v be r **patterned** from s to t (written $v = r|_s^t$). Then, $v \in \mathcal{R}(A,B)$, where:

$$\nu_v(b) = \begin{cases} \{b\} & \lambda_t(b) \neq \varnothing \\ \{b\} \cup (\widehat{\nu}_v \circ \lambda_r(x)) & \lambda_t(b) = \varnothing \end{cases} \tag{19}$$
$$\lambda_v(a) = \widehat{\nu}_v \circ \lambda_s(a) \tag{20}$$
$$\phi_v(a,b) = \phi_s(a, \phi_r(b,b)) \tag{21}$$
$$\rho_v(a,b) = \rho_s(a, \rho_r(b,b)) \tag{22}$$

3.3 Properties of rules

Definition 3.3.1. Let $r \in \mathcal{R}(A,B)$. r is **repeatable** iff $A \supseteq B$.

Definition 3.3.2. Let $r \in \mathcal{R}(A,B)$. r is **passive** iff $\phi_r(a,b) = a$ for all $a \in A$.

Definition 3.3.3. Let $r \in \mathcal{R}(A,B)$ and $s \in \mathcal{R}(A,C)$. r and s are **independent** ($r \perp s$) iff, for all ($a \in A$, $b \in B$, $c \in C$), $\phi_r(\phi_s(a,c),b) = \phi_s(\phi_r(a,b),c)$.

4 Evaluators

5 Conclusion